Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Šandor Feldi

Interaktivna vizualizacija arhitekture

diplomsko delo

na visokošolskem strokovnem študiju
računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Interaktivna vizualizacija arhitekture

Tematika naloge:

Izberite in opišite izbrane tehnologije in orodja za razvoj aplikacije za interaktivno vizualizacijo arhitekture. Predstavite sorodne rešitve. Implementirajte osnovno funkcionalnost za čimboljšo potopitev v navidezni svet arhitekturne rešitve in jo ustrezno testirajte. Na koncu opišite uporabo izdelka.

Zahvaljujem se mentorju izr. prof. dr. Peter Peeru za nasvete in pripombe pri izdelavi tega dela. Zahvaljujem se Vladimirju Kocjančiču, ki je sodeloval pri testiranju izdelka. Zahvaljujem se vsem, ki ste me podpirali in spodbujali. Posebno se zahvaljujem moji dragi Maji, za pomoč, ideje, nasvete ter nenazadnje za izdelavo 3D modela idejne zasnove kulturnega središča v Mariboru, ki je rezultat njenega diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
3D	three dimensional	tridimenzionalno
AI	artificial intelligence	umetna inteligenca
2D	two dimensional	dvodimenzionalno
ASCII	American standard code for information interchange	ameriški standardni nabor za izmenjavo informacij
BCF	BIM collaboration file format	datotečni format za sodelovanje v BIM
BIM	builing information modeling	informacijsko modeliranje zgradb
C4D	Cinema 4D file format (Maxon)	datotečni format Cinema 4D (Maxon)
CAD	computer aided design	računalniško podprto oblikovanje
DAE	digital asset exchange file format (Collada)	datotečni format za izmenjavo digitalnih sredstev (Collada)
DWG	drawing file format (AutoCAD)	datotečni format za risbe (AutoCAD)
DXF	drawing exchange file format (AutoCAD)	datotečni format za izmenjavo risb (AutoCAD)
ECMA	european computer manufacturers association	evropsko združenje proizvajalcev računalnikov
FBX	filmbox file format (Kaydara)	datotečni format filmbox (Kaydara)
FPS	first person shooter	streljaška igra

GB	gigabyte	gigabajt
GUI	graphical user interface	grafični uporabniški vmesnik
IFC	industry foundation classes file format	datotečni format temeljnih razredov
IoT	Internet of things	Internet stvari
ISO	international organization for standardization	mednarodna organizacija za standardizacijo
LFS	large file support	podpora velikim datotekam
LOD	level of detail	raven podrobnosti
MB	megabyte	megabajt
MVP	minimum viable product	minimalno izvedljivi izdelek
OBJ	object file format (Wavefront Technologies)	datotečni format objekt (Wavefront Technologies)
RGB	red, green, blue	rdeča, zelena, modra
S3	simple storage service (Amazon)	enostavna storitev za shranjevanje
SRTM	shuttle radar topography mission	radarska topografija zajeta v nalogi raketoplana shuttle
SVN	subversion - versioning control system	subversion - sistem za upravljanje z izvorno kodo
USD	united states dollar	ameriški dolar
VR	virtual reality	navidezna resničnost

Povzetek

Naslov: Interaktivna vizualizacija arhitekture

Vizualizacija v arhitekturi je ključnega pomena, saj je bistven način komunikacije med arhitekti in naročniki oziroma uporabniki. Običajno se uporabljajo slike ali posnetki upodobitev, ki se lahko generirajo ure in ure, končni uporabnik pa lahko rezultat le gleda. Vedno zmogljivejši računalniški sistemi, bodisi namizni bodisi mobilni ter razmah prosto dostopnih zelo naprednih in zmogljivih pogonov za igre, ponujajo možnosti za izdelavo realnočasovnih, realističnih in interaktivnih aplikacij. Diplomsko delo predstavi razvoj aplikacije za interaktivno vizualizacijo arhitekture, idejne zasnove novega kulturnega središča v Mariboru, ki je izdelana z uporabo pogona za izdelavo iger, Unity3D. Uporabnik lahko v izdelani aplikaciji spreminja zunanje in notranje svetlobne pogoje ter tako opazuje, kako le-ta vpliva na arhitekturo. Prikazani so tudi zelo osnovni agenti (ljudje in vozila), ki se sprehajajo po in okoli zgradbe ali vozijo po prometnicah v neposredni bližini. Da bi bili dostopni čim večjemu krogu uporabnikov, smo želeli podpreti čim več platform, vendar smo se zaradi količine dodatnega dela in zaradi mnogo zmogljivejše opreme omejili le na namizne sisteme (OSX, Microsoft Windows, Linux).

Ključne besede: računalniška grafika, vizualizacija, interaktivnost, arhitektura, 3D, Unity, SRTM, kulturno središče, navidezna resničnost.

Abstract

Title: Interactive visualization of architecture

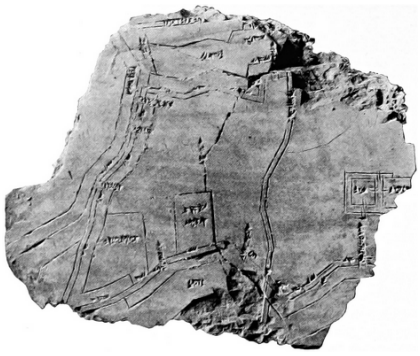
Visualization in architecture is crucial, because it is an essential way of communication between architects and clients or users. Normally, images or video renders are used, which can take hours to generate a visualization and all the end user can do is to watch it. Ever more powerful computer systems, whether desktop or mobile, and the availability of free very advanced and powerful game engines offer us possibilities to create real-time, realistic and interactive applications. The aim of this thesis is to create an application for interactive visualization of architecture, concept of a new cultural center in Maribor, using the Unity3D game engine. The user can manipulate the outside and inside lighting conditions, so it can observe how light interacts with the architecture. We also depict very basic agents (people and vehicles) that are walking in and around the building or are driving on the roads in the immediate vicinity. In order to be accessible to the widest range of users, we wanted to support many platforms, but due to the amount of extra work and hardware limitations of mobile devices we limited ourselves to desktop systems (OSX, Microsoft Windows, Linux).

Keywords: computer graphics, visualization, interactivity, architecture, 3D, Unity, SRTM, cultural center, virtual reality.

Poglavje 1

Uvod

Vizualizacija v arhitekturi je ključnega pomena, saj je bistven način komunikacije med arhitektom in naročnikom oziroma uporabniki. Prva znana vizualizacija arhitekture oziroma mesta, ki je bila narejena v pravilnem razmerju, izvira okoli 1500 let pred našim štetjem. Vklesana je v glineno tablico, prikazuje pa Mezopotamsko mesto Nippur (slika 1.1) [1].

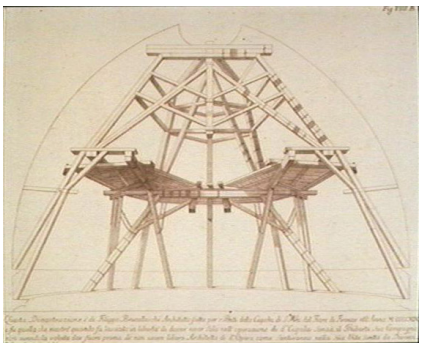


Slika 1.1: Mesto Nippur

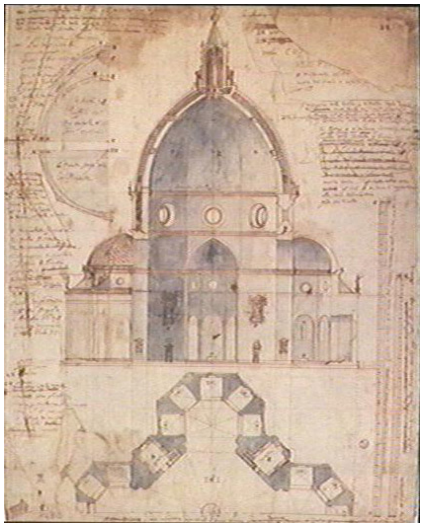
Slika 1.2: Mesto Rim

Vizualizacija je groba in prikazuje le ključne zgradbe ter obzidje. Bolj podrobna vizualizacija, ki vsebuje precej natančen tloris zgradb, je nastal v antičnem Rimu, med leti 203 in 211. Zemljevid velikosti $18,10 \times 13$ m vklesan v marmor, približno v merilu $1 : 240$, je prikazoval mesto Rim (slika 1.2) [2].

V srednjem veku so pri vizualizaciji zgradb začeli prikazovati tudi specifične podrobnosti na fasadah in notranjosti, ki so zgradbi dodali vrednost, bodisi estetsko bodisi funkcionalno, da bi lažje prepričali naročnike in upravičili visoke stroške gradnje. V renesansi smo dobili linearno perspektivo v umetnosti kot tudi v arhitekturi, pionir te tehnike je bil arhitekt Filippo Brunelleschi, uporabil jo je na prikazu kupole katedrale v Firencah (slika 1.3 in 1.4) [3]. Za 3D računalniško grafiko je linearna perspektiva ključnega pomena.



Slika 1.3: Gradnja kupole



Slika 1.4: Katedrala v Firencah

V sodobnem času se v arhitekturi velikokrat uporabljajo fizični modeli iz kartona, lesa ali plastike (3D tiskanje), ki niso vedno dovolj prepričljive ter se jih ne da hitro prilagajati željam in zahtevam naročnikov, posledično je vizualizacija sprememb okorna in neučinkovita. S pomočjo računalniških programov, kot so VRay, Mental Ray je možno izdelati zelo prepričljive (realistične) predstavitve, vendar so to statične slike, ali pa zgolj animacije, na katere uporabnik nima vpliva in jih lahko zgolj pasivno opazuje in sledi predstavitvi, kot si jo je zamislil avtor.

Ker je z uporabo 3D pogonov, kot so UnrealEngine, Unity3D mogoče izdelati zelo prepričljive, interaktivne, realnočasovne predstavitve arhitekture, pri katerih lahko dobimo takojšnje odzive uporabnikov, sem bil motiviran, da izdelam program za interaktivno vizualizacijo arhitekture. Uporabnikom omogočimo, da se lahko po lastni želji svobodno premikajo po okolici in prostorih, spreminjajo svetlobne razmere v prostorih, tako da prižigajo in ugašajo luči, nastavljajo poljuben čas dneva in opazujejo premikanje sonca ter senc, lahko premikajo notranjo opremo in še marsikaj. K predstavitvi bomo dodali okoliški cestni promet, ter osnovi prikaz gibanja ljudi po arhitekturi in okolici. Predstavitev bo na operacijskem sistemu Windows omogočila uporabo VR opreme, kot je Oculus Rift. Dodana vrednost rešitve je možna podpora široki paleti naprav (iOS, OSX, Android, Windows, Linux, WebGL itn.)

V poglavju 2 je predstavitev in primerjava nekaj sorodnih rešitev, ki se uporabljajo za arhitekturno vizualizacijo. V poglavju 3 so predstavljene uporabljene tehnologije in orodja pri razvoju aplikacije. V poglavju 4 je opisan izdelek. V poglavju 5 je predstavljen razvoj aplikacije, nato pa v poglavju 6 sledijo primeri uporabe in v poglavju 7 zaključek.

Poglavje 2

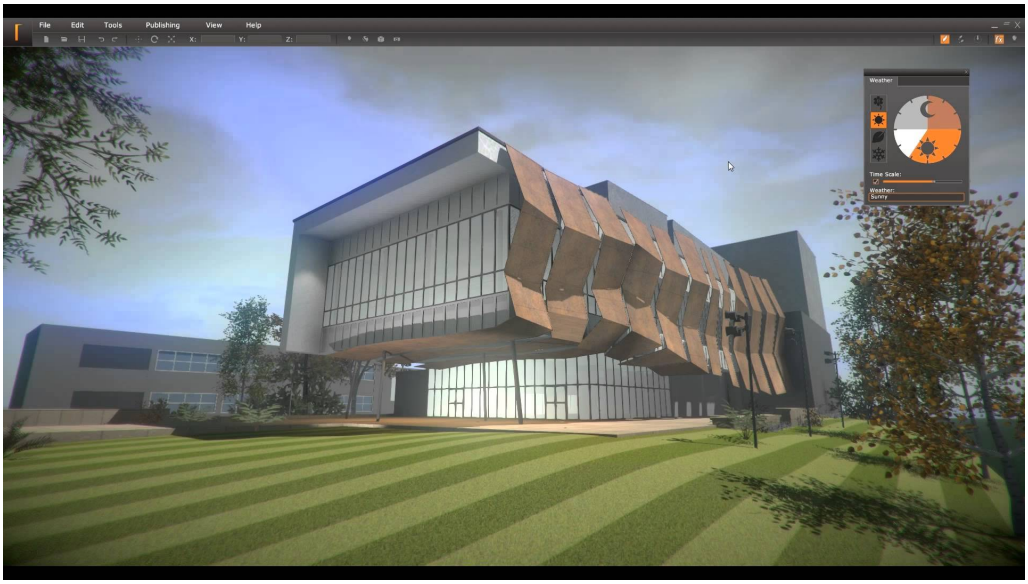
Sorodne rešitve

V tem poglavju je predstavljenih nekaj obstoječih rešitev, ki rešujejo problem interaktivne vizualizacije v arhitekturi. Večina rešitev je izdelana z uporabo pogona za izdelavo iger, Unity3D. Kljub temu da Unity3D podpira, v času pisanja te diplomske naloge, že 24 različnih platform (slika 2.1), pa je večina predstavljenih rešitev izdelana zgolj za eno.

Slika 2.1: Podprte platforme v Unity3D

2.1 Realis3D

Rešitev vsebuje program za interaktivni pogled 3D modela (slika 2.2) ter program, ki omogoča urejanje lastnosti uvoženega 3D modela kot tudi celotne predstavitve. Program zna ob zagonu uvoziti poljuben 3D model v obliki (angl. format) FBX in OBJ. Podpira Oculus in 3dTV.



Slika 2.2: Vizualizacija s programom Realis3D

Podprta sta operacijska sistema Windows in OSX. Programa za interaktivni pogled 3D modela nam ni uspelo usposobiti, da bi deloval, urejevalnik pa je deloval brez večjih težav. Urejevalnik omogoča tri različne poglede v svet: prvoosebna hoja, letenje, orbitalni pogled 3D modela. Omogočeno je nastavljanje kakovosti izrisovanja senc in tekstur. Uporabnik lahko spreminja letni čas in uro v dnevu s pomočjo posebne nadzorne plošče, kar spreminja lego sonca in lune, s tem pa se tudi spreminja osvetljenost scene. Dodatno lahko uporabnik uravnava vremenske razmere kot na primer deževanje, megla, sneženje. Sneženje ni prepričljivo in ne spremeni lastnosti materialov na objektih.

Možna je hoja v zgradbi ter prižiganje in ugašanje luči iz stranskega menija. Upravljanje je malo nerodno. Možna je izdelava upodobljene (angl. rendered) slike oziroma posnetka in fotosfere. Upodobljena slika ni predstavljala trenutnega pogleda. Scena ni bila prepoznavna, ker je bila shranjena slika v celoti črna. Urejevalnik omogoča spreminjaje materialov (tekstur) ter dodajanje in premikanje objektov. Program daje občutek, kot da je še v fazi razvoja. Domača stran programa in podjetja ne obstaja več.

2.2 Vizzera

Je bil 3D načrtovalni sistem za zimske olimpijske igre 2014, ki so se odvijale v Sočiju (slika 2.3).

Slika 2.3: Vizualizacija s programom Vizzera

Podprta sta operacijska sistema Windows in OSX. Možni pogledi so prvoosebna hoja in letenje ter celostni pregled zemljevida sveta, z možnostjo izbire interesne lokacije, kamor se lahko teleportiramo. Podprt je tudi pogled tlorisov nekaterih objektov. Lahko izbiramo med dnevom in nočjo, vendar brez vmesnih prehodov. Ni možna izbira vremena. Dinamične sence delujejo samo izven zgradb. Vizualizacija prikazuje minimalni avtomobilski promet po cestah ter sprehajajoče pešce. Po zgradbi se lahko sprehajamo, tako da se določeni zgradbi približamo. Če se nam prikaže ikona za vstop in raziskovanje zgradbe, potem lahko s klikom nanj stopimo v poslopje. Od daleč ni mogoče razbrati, v katere zgradbe lahko stopamo. Pogled iz zgradb ni možen. Ni interakcije z vrati in objekti, lahko pa dodajamo nekaj osnovnih predmetov, kot na primer bankomate, mize, stole itn. v svet. Imamo možnost posneti in predvajati sprehod uporabnika skozi sceno, kar je koristno za avtomatizirano

vodenje po arhitekturi in okolici. Program ponuja tudi orodja za merjenje razdalj, dodajanje opomb kjerkoli na sceni. Lahko shranimo sliko trenutnega pogleda. Program v času pisanja ne obstaja več.

2.3 ArchVirtual

ArchVirtual je podjetje, ki se ukvarja z interaktivno vizualizacijo arhitekture po naročilu. Njihov istoimenski program (slika 2.4) je izdelan samo za operacijski sistem Windows.

Slika 2.4: Vizualizacija s programom ArchVirtual

Uporabniške kontrole so omejene na Xbox igralni pripomoček. Podprte so tudi VR naprave, kot je Oculus. Xbox igralnega pripomočka nimam, zato nadaljnji test ni bil možen. Iz specifikacij in video posnetka na spletni strani produkta je razvidno, da je projekt izdelan na osnovi Unity 5, ima podporo za Oculus Rift, ter da ponuja interaktivno okolje in večigralnost. Velika dodana vrednost je tudi prostorski zvok in je edini produkt, ki ponuja to funkcijo. Uporabnik lahko spreminja lastnosti materialov (teksture in barve) sten in tal, dodaja in premika objekte oziroma pohištvo, spreminja vreme in čas

dneva. Produkt ponuja tudi dinamično osvetljevanje, panoramske poglede in navigacijsko mapo. Je najbolj dovršen produkt od vseh naštetih.

2.4 LightUp

Je vtičnik za program Sketchup (slika 2.5), ki omogoča zelo osnovno prvoosebno premikanje po modelu, hkrati pa osvetljuje tudi v realnem času. Z manipulacijo materialov lahko dosežemo učinke, kot so odboji, lom svetlobe itn., ki povečajo realističnost. Končna upodobitev je slika ali video posnetek.

Slika 2.5: Urejanje modela s programom LightUp

2.5 Viso3D

Program je sestavljen iz dveh komponent. Prva komponenta je vtičnik za program Sketchup, druga komponenta je pregledovalnik za iOS naprave (slika 2.6). Vtičnik omogoča izvoz 3D modela, ki ga lahko pregledujemo na iOS napravi v prvoosebni perspektivi. Dinamičnega osvetljevanja ni, lahko pa se uporabi svetlobna mreža (angl. baked lightmap).

Slika 2.6: Vizualizacija s programom Viso3D na napravi iPod

2.6 Lumion

Lumion (slika 2.7) je zelo kakovostno in tehnično dovršeno orodje za izdelavo predstavitev arhitekturnih modelov. Deluje samo na operacijskem sistemu Windows. Končni produkt je upodobitev slike ali video posnetek. Interaktivno gibanje po svetu ni možno. Urejevalnik je daleč najbolj dokončan od vseh prej naštetih produktov, omogoča kakovostno in prepričljivo manipulacijo vremena (oblakov), ure, materialov, oblikovanje terena in vodnih površin, postavljanje objektov. S spremembo ure se primerno premikajo sence objektov. Vsebuje veliko zbirko modelov, kot so drevesa, animirane osebe, živali, vozila, zgradbe, ceste, pohištvo. Osebe in vozila lahko animiramo, da se premikajo na sceni.

Slika 2.7: Urejanje vizualizacije s programom Lumion Free 3.2.1

2.7 Tabelarična primerjava produktov

Za večjo preglednost, predstavimo primerjavo predhodno opisanih produktov v tabeli 2.1.

	Realis3D	**Vizzera**	**Arch-Virtual**	**LightUp**	**Viso3D**	**Lumion**
platforma	Win, OSX	Win, OSX	Win, Xbox	SketchUp Win, OSX	SketchUp Win, OSX	Win
program	da	da	da	ne	ne	da
vtičnik	ne	ne	ne	da	da	ne
urejeval-nik	da	ne	ne	da	ne	da

pregledovalnik	da	ne	ne	da	ne	ne
interaktivnost	da	da	da	da	da	ne
VR	Oculus	ne	Oculus	ne	ne	ne
pogled	prvo-osebni, leteči, orbitalni	prvo-osebni, leteči, zemljevid	prvo-osebni, panorama, zemljevid	prvo-osebni, leteči, orbitalni	prvo-osebni	leteči
vreme	dež, sneg, megla	ne	dež, sneg, megla	ne	ne	oblaki
poljubna ura	da	da	da	da	ne	da
letni čas	da	ne	ne	ne	ne	ne
postopni prehodi	da	ne	da	ne	ne	da
dinamične sence	da	samo izven zgradb	da	da	da	da
preklop luči	da	ne	ne	ne	ne	ne
sprememba materialov sten in tal	da	ne	da	da	ne	da

pohištvo, objekti	da	da	da	ne	ne	da
opombe	ne	da	ne	ne	ne	ne
merjenje razdalj	ne	da	ne	ne	ne	ne
promet	ne	da	ne	ne	ne	da
več-igralnost	ne	ne	da	ne	ne	ne
interesne točke	ne	da	ne	ne	ne	ne
snemalnik gibanja	ne	da	ne	ne	ne	ne
izvoz slik	da	da	ne	da	ne	da
izvoz videa	da	da	ne	da	ne	da
prostorski zvok	ne	ne	da	ne	ne	ne

Tabela 2.1: Primerjava sorodnih rešitev

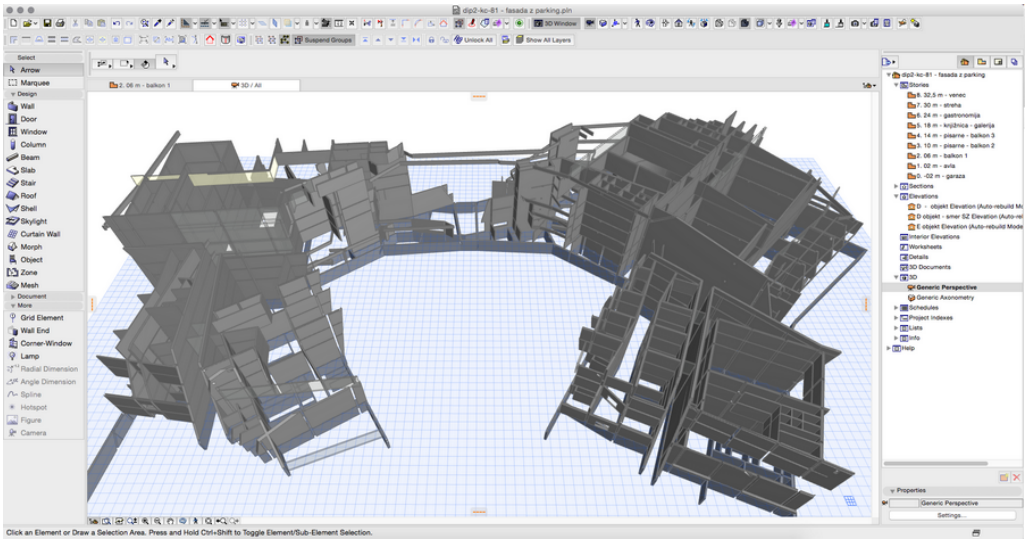
Poglavje 3

Uporabljene tehnologije in orodja

V tem poglavju opišemo pomembnejša orodja in tehnologije, ki so bile uporabljene pri razvoju tega projekta.

3.1 ArchiCAD

ArchiCAD je BIM/CAD programski paket za projektiranje v arhitekturi, katerega razvija podjetje Graphisoft. Deluje na operacijskih sistemih OSX in Windows. Omogoča učinkovito projektiranje v 2D in 3D načinu z uporabo parametričnih objektov. Deluje nad lastnim formatom PLN, ter podpira uvoz in izvoz v formate DWG, DXF, IFC in BCF. Pri svojem projektu sem ga uporabljal, da sem lahko 3D model zgradbe (slika 3.1), ki sem prejel od študentke arhitekture M. Hausmeister [4], po potrebi prilagodil in izvozil v format C4D za nadaljnjo obdelavo v programu Cinema4D.



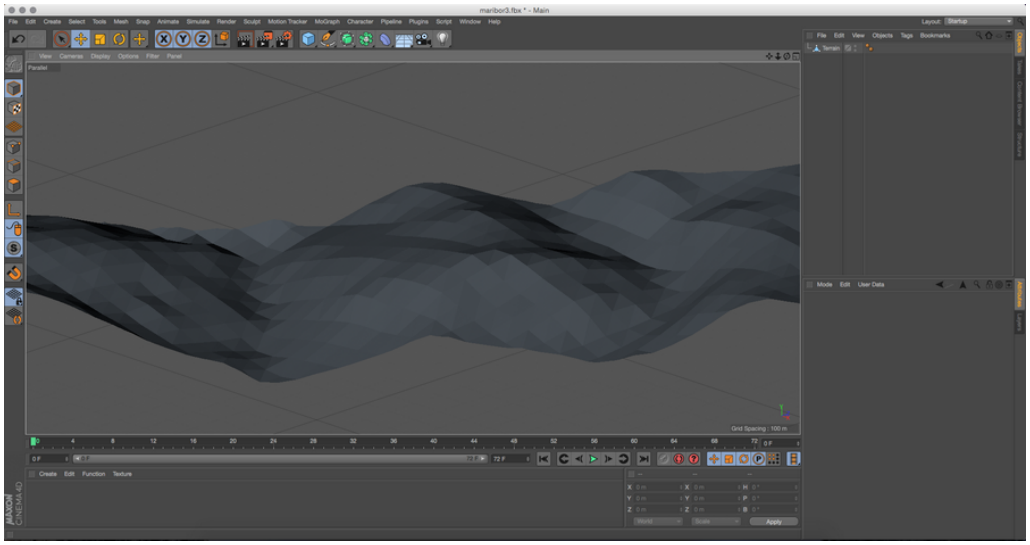
Slika 3.1: Grafični vmesnik ArchiCAD

3.2 C4D Exchange vtičnik

Ker so formati datotek, nad katerimi ArchiCAD deluje, za Unity3D praktično neprimerni, potrebujemo vtičnik C4D Exchange. Vtičnik omogoča izvoz 3D modela v formatu C4D, ki je primeren za nadaljnjo obdelavo v programu Cinema4D. Vtičnik za ArchiCAD je dostopen brezplačno na spletni strani podjetja Graphisoft.

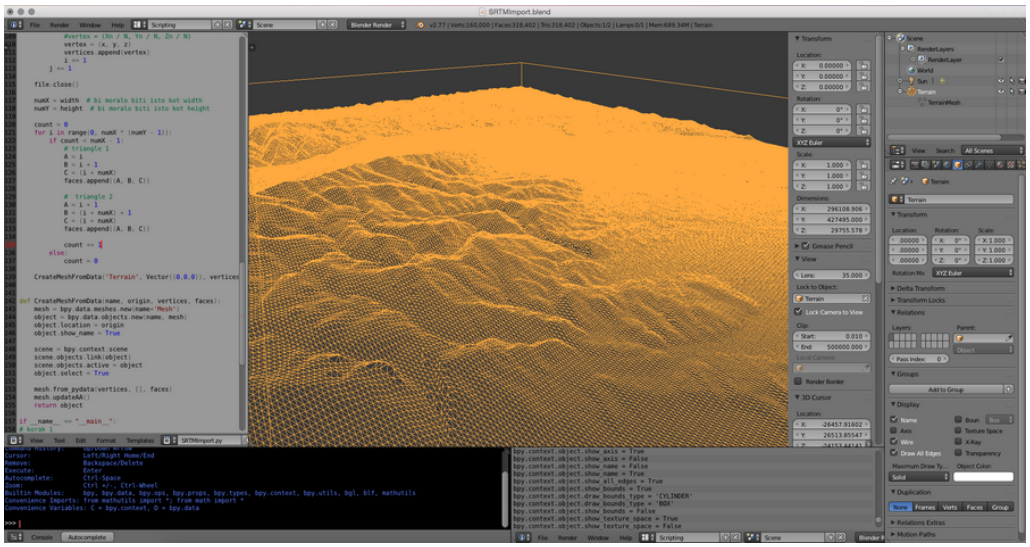
3.3 Cinema4D

Je program za 3D modeliranje, animacijo in upodabljanje (rendering). Deluje na operacijskih sistemih OSX in Windows. Na začetku je bil uporabljen zgolj za pretvorbo 3D modela zgradbe iz C4D v FBX format, ki je najbolj primeren za uvoz v Unity3D. Kasneje se je program izkazal kot zelo uporabno orodje pri dodatni obdelavi 3D modela zgradbe. Opravila kot na primer združevanje poligonov, optimizacija mreže, preusmerjanje normalnega vektorja ploskve in mehčanje digitalne elevacije (Sculpt→Subdivide) (slika 3.2).



Slika 3.2: Grafični vmesnik Cinema4D

3.4 Blender



Slika 3.3: Grafični vmesnik Blender

Blender (slika 3.3) je brezplačen odprtokodni program, namenjen 3D modeliranju in animaciji. Deluje na operacijskih sistemih OSX, Linux, Windows. Uporabil sem ga predvsem zaradi možnosti uporabe Python skript, s katerimi sem izdelal 3D model digitalne elevacije okolice. 3D modele zna izvoziti tudi v FBX formatu.

3.5 Unity 3D

Unity3D je zelo priljubljen pogon za izdelavo 2D in 3D iger, ki ga razvija podjetje Unity Technologies. Različica 5.4 pokriva že 24 platform (slika 2.1):

- spletne platforme: WebGL
- namizne platforme: OSX, Windows, Windows Store, Linux/SteamOS
- mobilne platforme: iOS, Android, Windows Phone, Tizen
- patforme za navidezno in agumentirana resničnost: Oculus Rift, Google Cardboard, Steam VR, Playstation VR, Gear VR, Microsoft Hololens
- igralne konzole: PS4, PlayStatiov Vita, XboxONE, Xbox360, Wii U, Nintendo 3DS
- pametni Televizorji: Android TV, tvOS, Samsung SMART TV

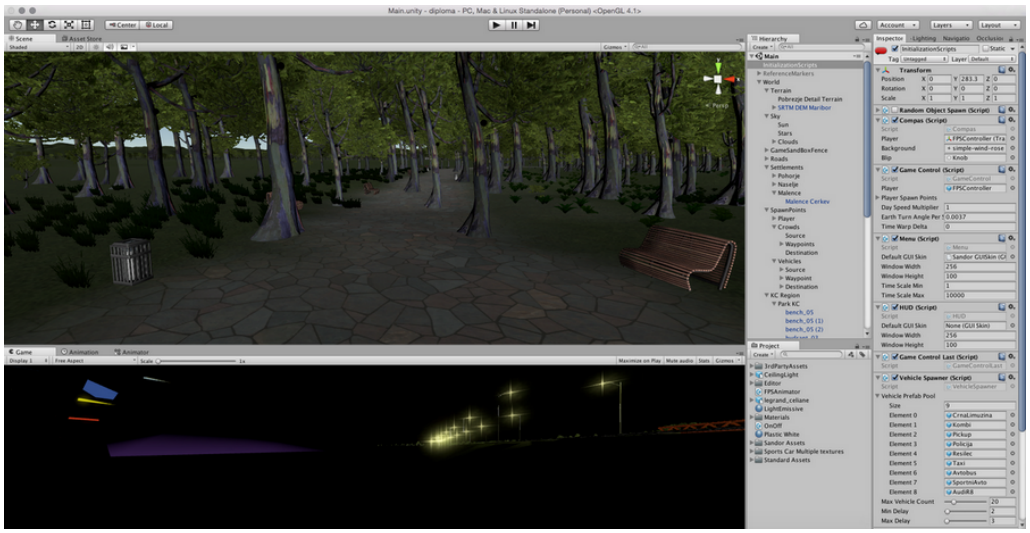
Programiranje je možno v jezikih C#, JavaScript in Boo. Privzeti urejevalnik kode in razhroščevalnik na operacijskem sistemu OSX je MonoDevelop, na operacijskem sistemu Windows pa Visual Studio Community. Unity3D je na voljo v brezplačni različici Personal, pod pogojem, da v finančnem letu ne zaslužimo z izdelkom, ki smo ga naredili v Unity3D, več kot 100.000,00 USD. Če presežemo to mejo, se moramo odločiti za eno od plačljivih naročnin: Plus, Pro ali Enterprise. Brezplačna različica ponuja vse zmogljivosti igralnega pogona. Bistvena razlika med brezplačno različica in naročniško licenco je v omejitvah pri dodatnih storitvah, kot so: podrobnost analitike (angl. Unity Analytics), prioriteta gradnje v oblaku (angl. Unity Cloud Build),

obseg večigralnosti (angl. Unity Multiplayer), poročanje o izjemah med izvanjem (angl. Unity Performance Rerporting) itn.

Uporabniški vmesnik Unity3D (slika 3.4) je intuitiven urejevalnik scene, s katerim lahko:

- uvažamo sredstva (angl. assets): 3D modeli, slike, zvočni posnetki, paketi drugih avtorjev (angl. Unity Package)
- postavljamo in urejamo primitivne 2D, 3D in uvožene kompleksne objekte
- dodajamo in urejamo svetlobne vire
- gradimo in spreminjamo lastnosti materialov
- gradimo uporabniški vmesnik iz osnovnih gradnikov itn.

Za Unity3D sem se odločil zaradi podpore mnogim aktualnim platformam, podpore programskemu jeziku C#, ki mi je od podprtih najljubši, dobre uporabniške podpore v forumih in ker ga lahko v svojem projektu uporabljam brezplačno.



Slika 3.4: Uporabniški vmesnik Unity3D

3.6 MonoDevelop

MonoDevelop je brezplačno odprtokodno razvojno okolje osredotočeno na razvoj projektov z ogrodji Mono in .NET. Uporabniški vmesnik ponuja podobne funkcije za urejanje (angl. refactoring, formatting) in zaključevanje programske kode (angl. autocomplete), integracijo s sistemi za upravljanje z izvorno kodo (angl. source control) itn., kot jih ima Microsoft Visual Studio. Deluje na operacijskih sistemih OSX, Windows, Linux. Uporabil sem ga za razvoj in razhroščevanje vse programske logike svojega projekta.

3.7 Python

Python je zelo razširjen in priljubljen skriptni jezik, ki je preprost in dobro berljiv. Python ima popolnoma dinamične podatkovne tipe, samodejno upravlja pomnilnik in podpira funkcionalno, imperativno oziroma proceduralno, strukturirano in objektno orientirano računalniško programsko paradigmo. Zelo priljubljen je pri podatkovnem rudarjenju, IoT projektih, grafiki. V svojem projektu sem ga uporabil za pripravo digitalne elevacije okolice iz izvornih podatkov in generiranje 3D modela v programu Blender.

3.8 .NET C#

C# je večparadigmski programski jezik, ki obsega močo tipizacijo ter imperativno, deklarativno, funkcijsko, generično, komponentno orientirano in objektno orientirano programiranje ter vsebuje zmožnost refleksije. Upravljanje s pomnilnikom temelji na čistilniku spomina. C# je razvilo podjetje Microsoft v okviru razvoja ogrodja .NET. Jezik sta kot standard odobrili organizaciji Ecma in ISO. Vsa programska logika mojega projekta je pisana v tem jeziku.

3.9 Bitbucket/Github

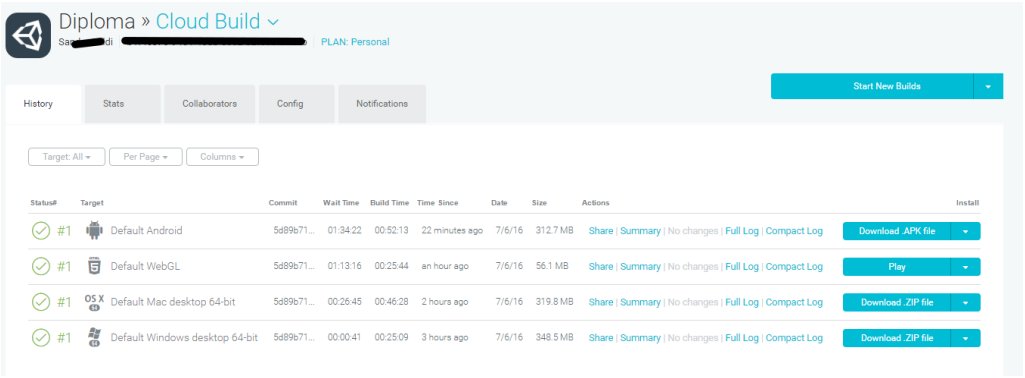
Bitbucket je spletni gostiteljski servis za repozitorije v Git. Ponuja distribuirano upravljanje z izvorno kodo, spletni grafični vmesnik, zagotavlja kontrolo dostopa in druge storitve za sodelovalni (angl. collaborative) razvoj. Prvotno sem ga za projekt izbral, ker ponuja neomejeno število brezplačnih zasebnih repozitorijev in zaradi seznanjenosti z okoljem, ker sem ga uporabljal tudi pri drugih projektih. Pri razvoju projekta arhitekturne vizualizacije se je hitro pokazala omejitev storitve Bitbucket. Sistem začne opozarjati uporabnike repozitorija, ko ta doseže velikost 1GB. Ko repozitorij doseže 2GB pa je popolnoma onemogočeno pošiljanje sprememb, ukaz git push, nanj. Projekti, kot so igre, oziroma so delani s pogoni za igre, lahko hitro dosežejo velikosti nekaj GB. Zaradi navedenih omejitev sem se odločil projekt preseliti na Github.

Github je konkurenčen in bolj priljubljen spletni gostiteljski servis za repozitorije Git. Zasebni repozitoriji so plačljivi, zato ga praviloma nisem uporabljal. Meni najbolj pomembna storitev Github je podpora LFS, ki je plačljiv, vendar omogoča shranjevanje in verzioniranje datotek zelo velikih datotek. Pri Github, moramo vse datoteke, ki so večje od 100MB, shranjevati z uporabo Github LFS. Velike datoteke se shranjujejo na oddaljene strežnike Amazon S3 spletnega servisa.

3.10 Unity Cloud Build

Unity Cloud Build (slika 3.5) je storitev, ki jo ponuja podjetje Unity Technologies. Omogoča avtomatizirano in hitro prevajanje, nameščanje in testiranje projekta na izbrane platforme v okviru razvojne ekipe. Storitev povežemo z Git ali SVN repozitorijem, kjer hranimo projekt. Unity Cloud Build periodično preverja ali je vsebina repozitorija posodobljena, če je, pripravi verzije projekta za vse izbrane platforme in pošlje obvestila preko elektronske pošte vsem uporabnikom, ki sodelujejo pri projektu. Brezplačna različica, čeprav ima najnižjo prioritetno vrsto, zelo skrajša čas prevajanja in izdelavo distri-

bucijskih paketov za vse izbrane platforme v primerjavi z ročno sprožitvijo na domačem razvojnem računalniku.



Slika 3.5: Unity Cloud Build

Poglavje 4

Opis izdelka

Program prikazuje idejno zasnovo kulturnega središča mesta Maribor in ožje okolice.

Program je zasnovan po vzoru prvoosebnih 3D iger, ki igralca potopijo (angl. immerse) v navidezno okolico, kar je ključnega pomena tudi za predstavitev arhitekture. K interaktivnosti programa pripomore intuitiven nabor uporabniških kontrol, ki je že postal ad-hoc standard v prvoosebnih igrah in predstavitvah. Uporabnik se lahko poljubno premika po okolici in po sami arhitekturi, upravlja stikala za luči, s čimer spreminja svetlobne pogoje v odsekih zgradbe, lahko pa tudi premika nekatere predmete, ki so postavljene v okolico in jih poljubno razporeja. Iz glavnega menija uporabnik lahko upravlja s ciklom dneva in noči, kar mu omogoča vizualizacijo interakcije svetlobe z arhitekturo in okolico. Da je predstavitev kar se da prepričljiva, smo za prikaz okoliškega terena (Pohorje, Dravska dolina, Slovenske gorice) uporabili brezplačen, a kakovosten nabor digitalne elevacije, kar pomaga uporabniku pri orientaciji in pri vizualizaciji pogledov iz zgradbe. Da celotna predstavitev zaživi, smo dodali vizualizacijo prometa na neposrednih prometnicah (Puhova ulica in avtocesta) ter vizualizacijo gibanja ljudi po parku in kulturnem središču.

Uporabnik lahko izbira izhodiščno točko oziroma točko interesa, ki je lahko zunaj zgradbe, ali pa je to neki prostor v zgradbi, kot na primer galerija,

kavarna, knjižnica, dvorana itn. Če mu je kakšen pogled prav posebno všeč, lahko zajame sliko, ki se shrani na trdi disk.

4.1 Podprte platforme

Program je bil preizkušen in prilagojen za namizne platforme, OSX in Windows ter za spletno platformo WebGL. V okolju Windows je omogočena podpora za Oculus Rift.

Podpora za mobilne platforme je predvidena, ni pa še uresničena, ker je potrebno prilagajanje na omejitve, ki jih te platforme imajo.

4.2 Uporabniške kontrole

Kontrole so klasične, kot jih poznamo iz prvoosebnih 3D iger za namizne računalnike.

Za spreminjanje smeri pogleda uporabljamo miško, za hojo pa kombinacijo tipk WASD oziroma smerne puščice. Sledi pregledni seznam tipk in akcij:

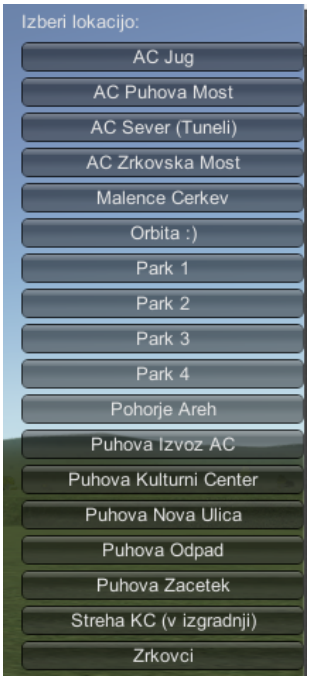
- W – korak naprej
- A – korak levo
- S – korak nazaj
- D – korak desno
- preslednica – skok
- levi shift – OSX, tek
- levi control – Windows, tek
- E – akcija (prižiganje in ugašanje luči, premikanje objektov)
- F – ročna svetilka

- Esc – glavni meni
- F12 – ekranska slika
- F1 – nalaganje trenutnega stanja igre
- F2 – shranjevanje trenutnega stanja igre

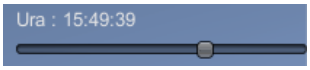
4.3 Glavni meni

Program vsebuje zelo preprost glavni meni, ki ponuja naslednje akcije:

- izbiro izhodiščne točke, ki so bile med razvojem vnaprej določene (slika 4.1)
- izbiro ure v dnevu (slika 4.2)
- spreminjanje hitrosti poteka dneva (slika 4.3)
- izhod iz programa/menija (slika 4.4)

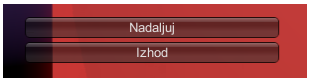


Slika 4.1: Meni, izbira izhodiščnih točk



Slika 4.2: Meni, izbira ure dneva

Slika 4.3: Meni, izbira hitrosti dneva



Slika 4.4: Meni, izhod iz programa/menija

Poglavje 5

Razvoj

V tem poglavju opišemo, kako je program nastajal. Kako smo pripravili sceno in objekte ter kako smo sprogramirali programske sklope. Nekatera podpoglavja vsebujejo tudi kratek teoretični uvod.

5.1 Izdelava scene

Sceno se primarno gradi grafično z urejevalnikom scene Unity. Vsak gradnik na sceni je tipa `GameObject`, ki je sestavljen iz komponent. Obvezna komponenta vsakega gradnika je `Transform`, ki določa položaj, vrtenje in velikost objekta na sceni. Ostale komponente so poljubne in se razlikujejo glede na vrsto in potrebe objekta, ki mu pripadajo. Gradnike na sceno postavljamo v hierarhično drevo (slika 5.1), kar nam izboljša preglednost nad sceno in omogoča preprosto manipulacijo celotnih vej v drevesu tako v urejevalniku kot tudi v naši programski kodi. Komponenta, v kateri je naša programska koda, se imenuje `Script` in je tipa `MonoBehaviour`, pripnemo jih lahko poljubno veliko na vsak objekt.

Za inicializacijo začetnih vrednosti in pripravo globalnega objekta edinca (angl. singleton) smo dodali prazen `GameObject` in ga poimenovali `InitializationScripts`. Razvoj programskega dela. Objekt `World` predstavlja naš navidezni svet, ki je dalje po funkcionalnih sklopih razdeljen na `Terrain`

(elevacija, rastje itn.), `Sky` (sonce, zvezde, oblaki), `GameSandBoxFence`, ki predstavlja fizične meje za prvoosebnega uporabnika, `Roads`, kjer imamo vse ceste in obcestne objekte, `Settlements`, ki predstavljajo nekaj okoliških naselij, `KC Region` je neposredna okolica kulturnega središča (centra) in sama glavna zgradba. `SpawnPoints` vsebuje samo navidezne objekte, ki se uporabljajo za izbiro začetnih lokacij oziroma navigacijo agentov (oseb in vozil).

Slika 5.1: Hirearhična zgradba scene

5.1.1 Priprava digitalne elevacije širše okolice

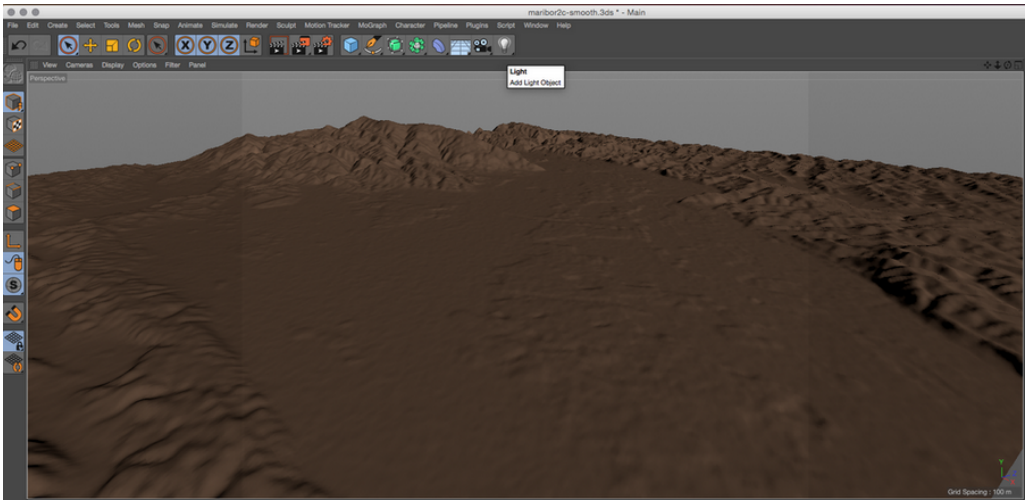
Projekt predstavlja in vizualizira idejno zasnovo novega kulturnega središča mesta Maribor [4]. Ker želimo doseči kar se da visoko stopnjo resničnosti in zaradi boljše orientacije v prostoru, moramo uporabniku prikazati prepoznavno okolico in referenčne geografske točke.

Naš cilj je bil prikazati, kar se da prepričljivo okolico Dravskega polja s pogledom na vzhodni del Pohorja ter zahodni del Slovenskih goric. Območje, ki je prisotno v projektu, se razprostira od 46.40N/15.30E do 46.20N/15.50E (zemljepisna širina/dolžina). V prvem poskusu smo uporabili preizkusno različico programa Sketchup, s katero se lahko izredno hitro zgradi digitalna elevacija na osnovi podatkov Google Maps, vendar smo ocenili, da je kakovost podatkov preslaba.

V drugem poskusu smo uporabili javno in brezplačno bazo podatkov projekta SRTM (angl. Shuttle Radar Topography Mission), različico V4.1 [5]. SRTM elevacijski model ima ločljivost 90 m pri ekvatorju. Podatki so na voljo kot mozaik 5×5 stopinj, ločljivosti 6000×6000 točk, ter so obdelani tako, da imajo kar se da malo anomalij, kot na primer špice, luknje, manjkajoči podatki. Točka v mozaiku predstavlja nadmorsko višino v metrih in je zapisana kot celo število. Vrednost -9999 pomeni, da podatek manjka.

Ploščica (angl. tile) za naš projekt je označena kot `SRTM` 40_03 in obsega območje od 45N/15E do 50N/250E (zemljepisna širina/dolžina). Podatki so zapisani v binarni GeoTiff formatu in v ASCII tekstovni datoteki. Uporabili smo preprostejšo ASCII datoteko. Za izvoz podatkov območja Dravskega polja in za polnjenje manjkajočih elevacijskih podatkov smo napisali preprosto Python skripto, ki je iz izvorne ASCII datoteke izluščila točke, ter jih zapisala v novo ASCII datoteko. V tem koraku smo poskrbeli tudi za interpolacijo in zapolnjevanje manjkajočih točk, ki so manjkale v eni vrstici. Ustrezal je preprosti izračun povprečja dveh točk, ki sta obdajali manjkajočo točko. Iz pripravljenih podatkov smo nato izdelali seznam točk (angl. vertex) in ploskev (angl. face), ki sta kot argument bila podana metodi `from_pydata`, ki je izdelala 3D model. Metoda `from_pydata` je del Blenderjeve Python knjižnice

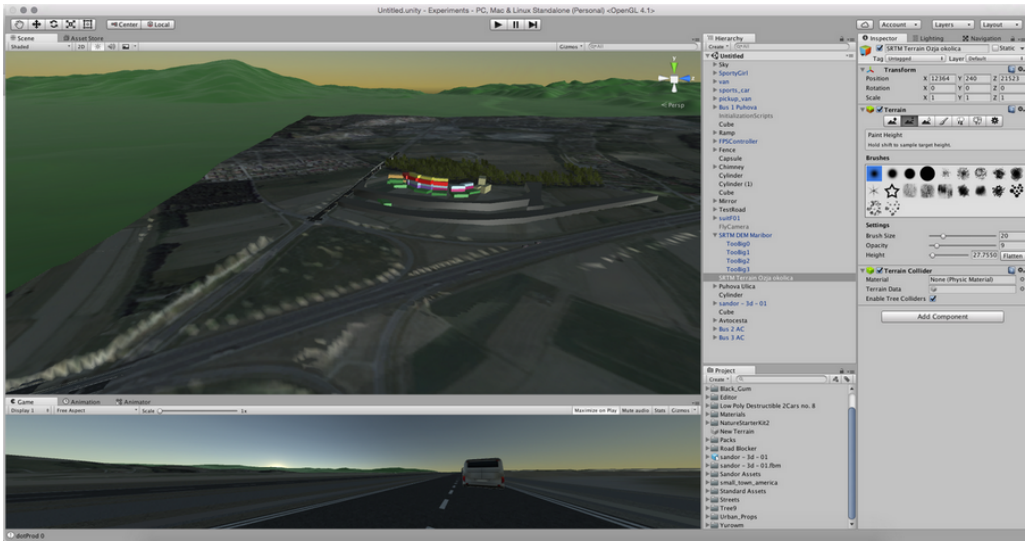
Data. Dobljeni 3D model okolice smo shranili v formatu DAE za nadaljnjo obdelavo v programu Cinema4D. Ker je 3D model imel ostre prehode med ploskvami, smo v Cinema4D izvedli ukaz Sculpt→Subdivide, kar je prehode naredilo veliko mehkejše, dovolj dobro in natančno za potrebe projekta. Pripravljen teren je prikazan na sliki 5.2.



Slika 5.2: Digitalna elevacija širše okolice

5.1.2 Priprava digitalne elevacije in urejanje ožje okolice

Pri izdelavi ožje okolice okoli kulturnega središča je bil uporabljen vgrajen Unity3D objekt Terrain, velikosti 1600×1600 m. Privzeto objekt predstavlja ravno ploskev velikosti 512×512 m.



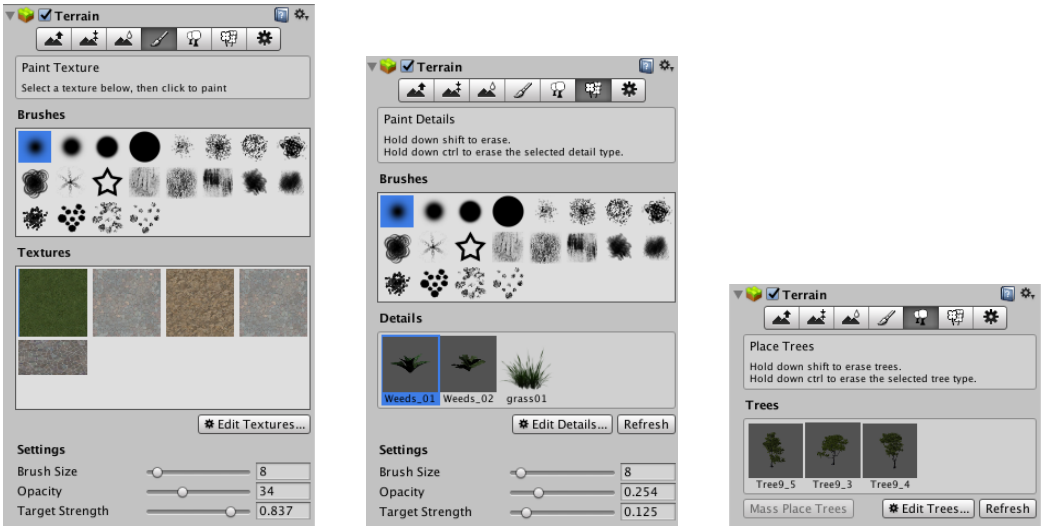
Slika 5.3: Urejanje ožje okolice

Teren lahko oblikujemo ročno s čopiči za urejanje višin, z uporabo slike, na kateri so višine upodobljene s sivinami v barvnem zapisu RGB, ali s pomočjo skripte. V prvem poskusu je bila uporabljena skripta, da smo z nastavljeno frekvenco odčitali višinske točke 3D objekta širše okolice, na območju, kjer se je ta prekrivala z objektom Terrain in to vrednost prepisali v objekt Terrain. Ker rezultati niso bili zadovoljivi, smo naredili nov objekt, višine pa smo urejali z vgrajenimi čopiči, kar je bilo časovno zamudno.

Objekt Terrain je del specializiranega pogona za teren (angl. terrain engine). Na teren lahko dodajamo objekte, ki jih potrebujemo v velikih količinah, kot na primer drevesa oziroma različno rastje, kot so trava, rože, grmovja, objekte, ki dodajo podrobnosti (angl. detail objects), kot na primer kamenje. Pri tako dodanih objektih na teren za raven podrobnosti objekta (angl. LOD - Level of detail) skrbi pogon za teren, upoštevajoč parametre, ki smo jih določili. Pomembno se je zavedati, da pri objektih, ki niso bili dodani v okolje posredno preko pogona za teren, prilagajanje ravni podrobnosti glede na oddaljenost ne deluje. Terenu lahko določamo tudi območja vetra (angl. wind zone), to so specializirani objekti, ki jim lahko določamo vrsto delovanja

(usmerjeno, sferično), radij, sunke, turbulenco. Ta objekt vpliva na krošnje dreves in sisteme delcev, za travo se veter nastavi posebej, v urejevalniku terena.

Za čim bolj natančno postavljanje gozdov, cest, naselij in drugih objektov je bil uporabljen satelitski posnetek okolice (slika 5.3, zgoraj), ki je bil na teren dodan kot tekstura. Višina terena, postavitev dreves, trave, grmovlja in teksture tal (trava, zemlja, tlakovci itn.) se rišejo s čopiči (slika 5.4), ki jim lahko nastavljamo lastnosti (gostota, radij itn.). Narejenih je bilo kar nekaj ponovitev (angl. iteration) terena.



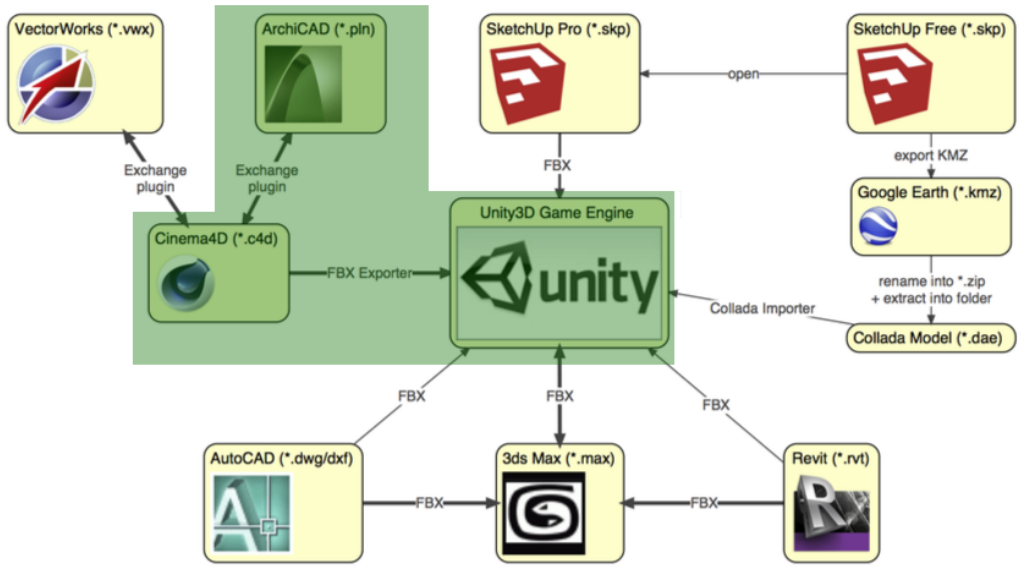
Slika 5.4: Čopiči in nastavitve za postavljanje objektov in tekstur na teren

Satelitski posnetek je bil po končanem postavljanju objektov odstranjen, ker ni bil zadovoljive kakovosti.

5.1.3 Postopek priprave objekta za uvoz v Unity3D

3D model oziroma kulturno središče je bilo izdelano v programu ArchiCAD. Ker ne moremo neposredno uporabiti 3D modela, ki ga dobimo v programu ArchiCAD, je le tega potrebno s pomočjo vtičnika C4D Exchange, shraniti v formatu C4D. To datoteko potem lahko uvozimo v program Cinema4D,

3D model pa shranimo v format FBX, ki je primeren za uvoz v Unity (slika 5.5) [6].



Slika 5.5: Potek dela (angl. workflow) priprave 3D modela za uvoz v Unity3D

Med delom se je izkazalo, da je treba na modelu popraviti še kakšno malenkost, kot na primer odpraviti nevidne poligone, tako da jim obrnemo vektorje normal ali združiti in optimizirati skupino poligonov, odstraniti neželene objekte, zamenjati ali nastaviti materiale (teksture). To delo je potekalo v Cinema4D.

5.2 Razvoj programskega dela

5.2.1 Struktura aplikacije

Unity programi so sestavljeni iz številnih skript, ki podedujejo lastnosti razreda `MonoBehaviour`. Take skripte lahko neposredno pripnemo na poljubne objekte na sceni, vrstni red njihovega izvajanja pa običajno ni določen. Hitro se zgodi, da imajo objekti pripetih več skript, ki izvajajo različna specializirana opravila. Če želimo dostopati do javnih spremenljivk in metod razreda

iz poljubne skripte, ki je pripeta na objekt, moramo dobiti njeno referenco, kar storimo z metodo `GetComponent`. Ker je tak način zelo neučinkovit in ker želimo nekatere parametre programa imeti dostopne na globalni ravni, smo uporabili načrtovalski vzorec edinec (angl. singleton), ki zagotavlja, da imamo na voljo natanko eno instanco določenega razreda. Naš razred je poimenovan `GameControl`.

5.2.2 Edinec GameControl

Edinec `GameControl` je hrbtenica programa, ki skrbi za stanje programa oziroma stanje objektov, ki so globalni in morajo biti hitro in kjerkoli dostopni za nadaljnjo obdelavo, kot na primer trenutni čas v programu, stanje igralca (naziv, zdravje, aktivna kamera, lokacija in orientacija), seznam interesnih točk itn. Stanje lahko shranimo na disk v binarno datoteko, ter ga kasneje tudi preberemo.

Stanje shranjujemo z uporabo mehanizma serializacije podatkov v programskih objektih oziroma podatkovnih strukturah, ki v našem primeru pretvori objekte v binarni niz. Za branje stanja uporabimo mehanizem deserializacije, ki iz binarne oblike podatkov izgradi programske objekte oziroma podatkovne strukture. Serializiramo lahko osnovne tipke podatkov, kot na primer `string, int, float, decimal`, itn. ter objekte, ki so sestavljeni iz osnovnih tipov podatkov, pod pogojem, da so označeni z atributom `System.Serializable`. To običajno velja za objekte, ki smo jih naredili mi. Za objekte, ki si specifični za Unity3D, kot na primer `Vector3` in `Quaternion`, neposredna serializacija ni mogoča, zato smo morali izdelati preprosta razreda `Vector3Serializable` in `QuaternionSerializable`, ki sta označena kot `Serializable` in hranita javno dostopne spremenljivke prej omenjenih Unity3D razredov.

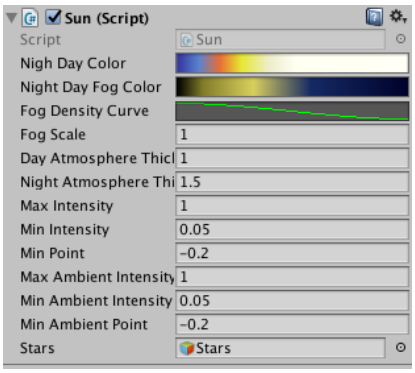
5.2.3 Prvoosebna kamera in premikanje

Za prvoosebno (angl. first person) kamero in premikanje smo uporabili priloženi Unity paket sredstev (angl. Unity asset package) ter v projekt uvozili sredstvo (angl. asset) `FirstPersonCharacter`. To montažno sredstvo (angl. prefab asset) je hierarhično sestavljeno iz dveh objektov: `FPSController` in `FirstPersonCharacter`. Pomembnejše komponente, ki jih ima `FPSController` privzeto pripete so: togo telo (angl. `Rigidbody`), krmilnik lika (angl. `Character Controller`) in skripto prvoosebni krmilnik (angl. `First Person Controller`), ki bere stanje tipk in miške ter krmili komponento `Character Controller`. Ker sta si komponenti `Rigidbody` in `Character Controller` medsebojno izključujoči, nam pa je bolj pomembno, da lahko lik upravljamo z miško in tipkovnico, kot da nanj vplivajo fizikalne sile, smo odstranili `Rigidbody`. Kljub temu, lahko še vedno na druge objekte, ki vsebujejo komponento `RigidBody`, vplivamo s silo, tako da uporabimo funkcijo `OnControllerColliderHit()`, ko se trknemo z objektom. Spremenili smo tudi skripto `First Person Controller` in omogočili izbiro tipke, ki krmili tek, glede na operacijski sistem na katerem teče končni produkt.

5.2.4 Sistem dneva in noči

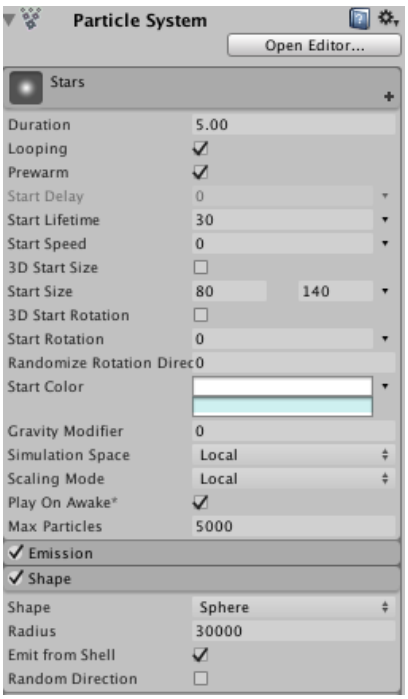
Sistem dneva in noči krmilita dve skripti, ki sta bili poimenovani `Sun` (Sonce) in `StarrySky` (zvezdno nebo). Prva je pripeta na objekt `Sun`, ki je preprosta usmerjena luč (angl. directional light), druga pa na objekt `Stars` (zvezde), ki vsebuje komponento sistema delcev (angl. particle system) v obliki krogle, ki oddaja delce na plašču krogle. Ker je sonce usmerjena luč, njeno izhodišče ni pomembno, vedno enako vpliva na okolico. Središče objekta zvezde pa smo postavili na lokacijo kulturnega središča in se ne premika z igralcem, tako kot bi se nebesna škatla (angl. skybox). Pripravili smo tudi lastno proceduralno nebesno škatlo ter gradiente za barvo neba in megle ter krivuljo gostote megle. Skripta `Sun`, krmili vrtenje luči `Sun`, računa skalarni produkt med soncem in navpičnico, ki ga omejimo med 0 in 1. Vrednost dalje uporabimo

za izračun jakosti glavne luči sonca in prostorske (angl. ambient) svetlobe ter da izluščimo vrednosti iz gradientnih objektov (slika 5.6), da dobimo barvo neba in gostoto atmosfere.



Slika 5.6: Parametri in gradienti skripte za krmiljenje prehoda dneva in noči

Skripta poskrbi tudi za vklop in izklop cestnih luči in zvezd ter za pravilno osvetlitev glede na izbrano uro in hitrost dneva. Skripta StarrySky skrbi za vrtenje sistema delcev zvezd (slika 5.7).

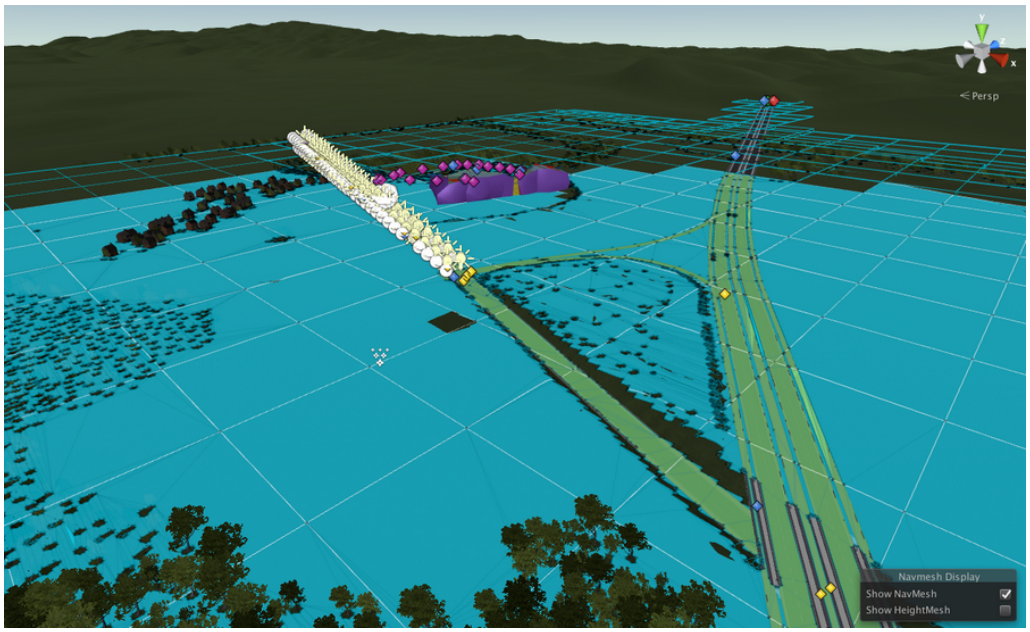


Slika 5.7: Sistem delcev zvezd

5.2.5 Navigacija vozil in ljudi

Navigacija je sposobnost agenta, v Unity3D je to objekt, ki ima pripeto komponento `NavMeshAgent`, da najde ustrezno pot do želenega cilja v navideznem okolju.

Unity navigacijski sistem uporablja navigacijske mreže (angl. navigation mesh), ki jih ustvarimo na osnovi geometrije scene. Navigacijska mreža je zbirka 2D konveksnih poligonov, ki določa območja, po katerih se agenti lahko premikajo. Sosednji poligoni so med seboj povezani v graf, optimalna pot pa se računa s pomočjo algoritma A*, A zvezda (angl. A star). Ta algoritem je zelo priljubljen v računalniških igrah zaradi svoje hitrosti in natančnosti [7].



Slika 5.8: Prikazuje področje navigacijske mreže - z zeleno za vozila, z modro za ljudi

V Unity3D lahko določimo do 31 navigacijskih mrež, ki jih omejimo na izbrane objekte oziroma področja, ter jim določimo ceno poti (angl. cost) med vozlišči. Tako smo ločeno določili navigacijsko mrežo za vozila, ki se drži objektov ceste, in navigacijsko mrežo za ljudi, ki se lahko gibljejo po terenu

in po zgradbi (slika 5.8 zgoraj).

Vsi montažni objekti (angl. prefabricated objects), ki se avtonomno gibljejo, morajo imeti pripeto komponento `NavMeshAgent`. Tej komponenti lahko določimo navigacijsko mrežo, ki jo uporablja, lastnosti agenta, kot so radij, višina, hitrost ter druge parametre. Objektom, ki niso statični, torej jih lahko premaknemo, vendar predstavljajo oviro in jo morajo agenti zaobiti, moramo pripeti komponento `NavMeshObstacle`, ki ji lahko določimo radij, višino. `NavMeshObstacle` komponento smo pripeli tudi na montažni objekt `FPSController`, ki predstavlja igralca, da ga agenti obidejo, če jim stoji na poti.

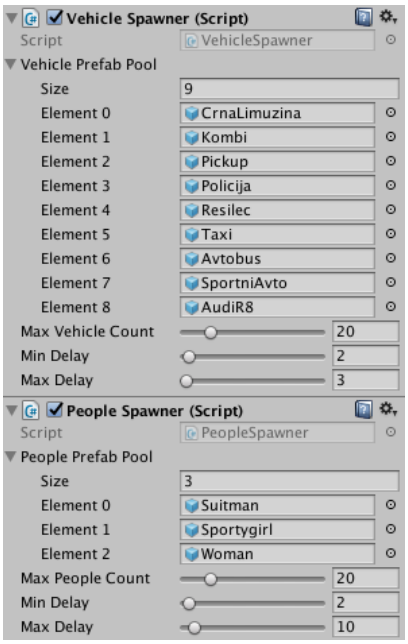
Na sceni so na ključnih lokacijah postavljeni prazni objekti (`GameObject`), ki predstavljajo začetne, vmesne in končne ciljne točke agentom. Na sliki 5.8 so predstavljene z ikonico romba, in sicer:

- vijoličaste so točke poti ljudi, ki so krožne
- zelene so začetne točke poti vozil, z oznako `VehiclesSpawn`
- rumene so vmesne točke poti vozil, z oznako `VehicleWaypoint`
- rdeče so končne točke poti vozil, z oznako `VehicleDestination`

Celotno pot hranimo kot seznam teh točk. Za premikanje vozil skrbi skripta `DriveTo`, za premikanje ljudi pa `MoveTo`, ki sta si trenutno zelo podobni. Glavna razlika je v tem, da `DriveTo` gre iz izhodiščne točke skozi vmesne in konča pot v končni točki, `MoveTo` pa vedno ponavlja določeno pot, tako kot so določene ciljne točke: krožno, od A do B in nazaj v A. `MoveTo` vsebuje tudi klice funkcij, ki krmilijo animacijo lika.

Da lahko upravljamo agenta, moramo najprej dobiti kazalec na komponento `NavMeshAgent`, kar storimo s klicem funkcije `GetComponent<NavMeshAgent>()`. Da agentu določimo novo ciljno točko na poti, pa uporabimo funkcijo `agent.SetDestination(objektTocka)`, ki ji, kot argument podamo prej navedeni prazni `GameObject`. Funkcijo `SetDestination` kličemo ob inicializaciji in nato vsakič, ko agent prispe do trenutno nastavljene ciljne točke.

Da agente postavimo na sceno, skrbita skripti `VehicleSpawner` in `PeopleSpawner`, ki sta pripeti na objekt `InitializationScripts` (slika 5.9).

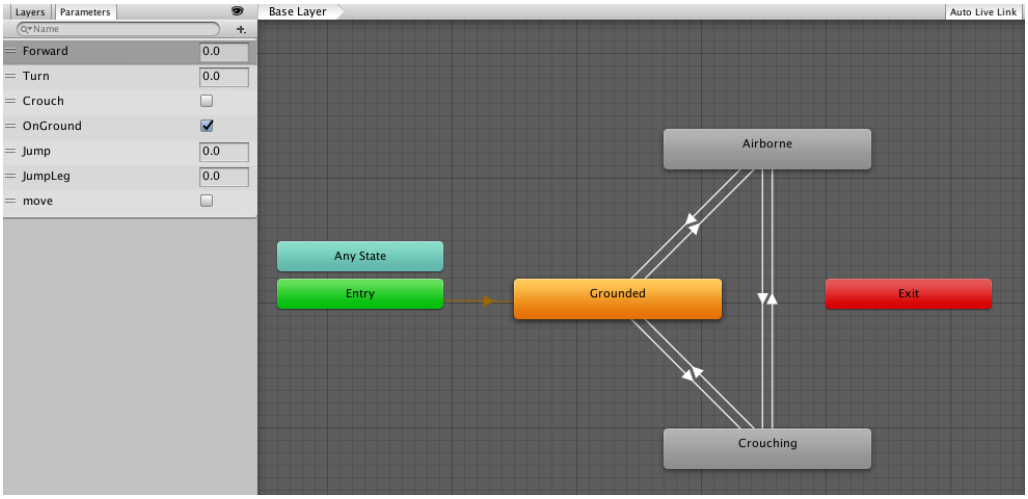


Slika 5.9: Parametri skript za generiranje vozil in ljudi

Obe skripti hranita seznam prednastavljenih poti in seznam montažnih objektov, ki jih naključno generirata v nekem časovnem intervalu. Montažni objekti so na primer različni modeli vozil in ljudi. Skriptama lahko nastavimo maksimalno število objektov, ki jih lahko postavita na sceno, ter časovni interval generiranja v sekundah (slika 5.9). Objekti se po končani poti, onemogočijo, tako da niso več aktivni na sceni in nato naključno reciklirajo in ponovno začnejo z izvedbo poti, ki jim je bila dodeljena.

5.2.6 Animacija likov

Unity vsebuje sistem za animacijo objektov, ki se imenuje Mecanim. Animacija objekta je predstavljena kot končni avtomat (angl. state machine), v katerem vsako stanje predstavlja končno animacijo (slika 5.10), ki je lahko enostavna ali sestavljena.



Slika 5.10: Končni avtomat animacij

Sestavljene animacije dobimo z uporabo mešalnega drevesa (angl. blend tree), iz katerega se vejejo osnovne animacije. Končnemu avtomatu lahko pripišemo parametre, ki jih lahko krmilimo iz skripte. S parametri definiramo pogoje, ki nam služijo pri prehodih iz enega stanja v drugega, s tem pa tudi iz ene animacije v drugo. Vsak končni avtomat v Unity3D vedno vsebuje stanja `Entry`, `Exit` in `Any State`. Vozlišče `Entry` se uporablja, ko aktiviramo avtomat, ter glede na stanje vhodnih parametrov, v njej določimo, v katero nadaljnje vozlišče preide avtomat. Vozlišče `Exit` se uporabi, ko želimo naznaniti, da se bo avtomat končal. Ni nujno, da se, zato ni vedno povezave na to vozlišče, običajno pa se stanje uporabi, ko so končni avtomati gnezdeni. Vozlišče `Any State` se uporabi, ko želimo preiti v neko novo stanje ne glede na trenutno stanje, v katerem je avtomat. Vsak avtomat mora imeti eno privzeto stanje, ki je na grafu označen v oranžni barvi [8].

Uporabili smo animacijske posnetke (angl. animation clips) za humanoidne like, ki jih dobimo kot dodatna sredstva (angl. asset) ob namestitvi Unity3D. Končni avtomat povežemo z montažnim objektom, v našem primeru z humanoidnim likom, tako, da na objekt pripnemo komponento Animator. Animatorju moramo določiti skelet (angl. avatar), ki predsta-

vlja sklepe in kosti objekta ter krmilnik (angl. controller), ki ga povežemo s končnim avtomatom. Skelet je za vsak objekt drugačen in se običajno samodejno gradi ob uvozu objekta v Unity3D [9].

V skripti `MoveTo` in `ThirdPersonCharacter`, ki sta pripeti na vsak montažni objekt, ki predstavlja človeka, krmilimo parametre končnega avtomata. To naredimo tako, da podamo želeni vektor hitrosti agenta funkciji `ThirdPersonCharacter.Move`, ki pretvori vektor iz globalnega prostora v lokalnega ter ga preslika na tla. Vrednosti preslikanega vektorja uporabimo za parametra končnega avtomata: `m_TurnAmount`, `m_ForwardAmount`.

5.2.7 Izhodiščne točke

Izhodiščne točke na sceni so predstavljene kot prazni objekti z oznako (angl. tag) `Respawn` in so v urejevalniku prikazane z modro ikono romba. Točke se izključno uporabljajo za igralca, kot referenčne točke, katerih vrednosti položaja in vrtenja se zapišejo v komponentro preobrazbe (angl. transformation) igralca, kar povzroči premik na izbrano točko.

Izhodiščne točke določimo grafično v urejevalniku ter jim dodelimo oznako `Respawn`. Imamo jih lahko poljubno veliko. Seznam možnih izhodiščnih točk programsko dobimo tako, da poiščemo vse objekte na sceni z oznako `Respawn` s funkcijo `GameObject.FindGameObjectsWithTag(\Respawn")`. Zaradi učinkovitosti in hitrosti to naredimo samo enkrat, in sicer v globalnem edincu `GameControl`.

Seznam izhodiščnih točk prikažemo v glavnem meniju kot seznam gumbov.

5.2.8 Uporabniški vmesnik

V Unity 5.4 lahko gradimo uporabniški vmesnik na dva načina: grafično z uporabo specializiranih objektov (angl. `GameObject`) za uporabniški vmesnik, kot so `Canvas`, `Text`, `Button`, ki jih razporedimo v hierarhično drevo, ali programsko, v neposrednem načinu (angl. immediate mode GUI), z upo-

rabo metod razredov `GUI` in `GUILayout`. Metode razredov `GUI` in `GUILayout` moramo klicati v `MonoBehaviour` funkciji `OnGUI()`. `GUILayout` omogoča samodejno postavljanje elementov uporabniškega vmesnika enega za drugim. Razred `GUI` se uporablja za ročno postavljanje elementov uporabniškega vmesnika, položaj pa se določa v prvem argumentu metod razreda [10].

Za prikaz trenutne ure in glavnega menija smo uporabili razred `GUILayout`, za prikaz kompasa pa razred `GUI`. Samodejna postavitev (angl. automatic layout) ima svoj začetek in konec, zato se vsaka postavitev začne z metodo `BeginArea(x, y, širina, višina)` in konča z `EndArea()`, vmes pa je poljubno število metod, ki izrisujejo besedilo, drsnike, gumbe, slike.

Vrednosti, kot so ura, množitelj ure, seznam izhodiščnih točk preberemo iz javnih spremenljivk globalnega edinca `GameControl`. Prav tako v javne spremenljivke pišemo, ko v glavnem meniju premikamo drsnike za uro in množitelj ure.

Funkcionalnost glavnega menija smo prikazali v poglavju 4.3.

Za izpis namigov, ko se zadenemo ob ali približamo objektu na sceni, kot na primer stikalu za luč, smo uporabili uporabniški vmesnik s specializiranimi objekti na sceni (angl. `GameObject`), `Canvas` in `Text`.

5.3 Testiranje

Testiranje izdelka je potekalo s pomočjo uporabe brezplačne storitve Unity Cloud Build, kjer lahko za vsak projekt navedemo vir izvorne kode, platforme za katere želimo graditi različice programa ter sodelujoče osebe, v našem primeru testerje, na projektu. Storitev periodično preverja ali je prišlo do sprememb na repozitoriju, če zazna, da smo naredili spremembo na projektu, zgradi različice za vnaprej določene platforme ter preko elektronske pošte obvesti testerje. Testerji si nato lahko zadnje različice programa prenesejo ter začnejo izvajanje testa glede na navodila, ki so jih prejeli. Cikel smo 5 krat ponovili s tremi testerji, dokler nismo bili zadovoljni s končnim rezultatom.

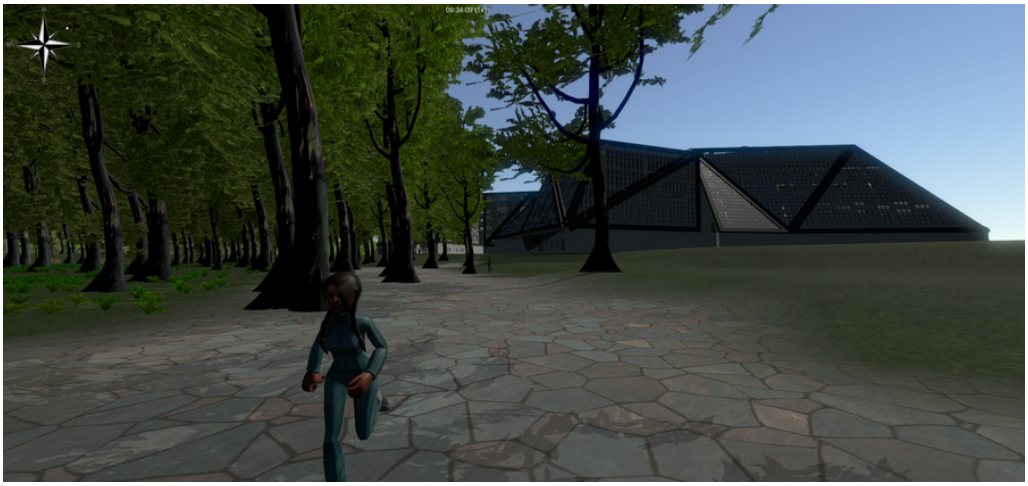
Poglavje 6

Uporaba izdelka

Da prikažemo namembnost programa, v nadaljevanju predstavimo našo rešitev ter nekaj primerov uporabe, ki smo jih predvideli v primeru nadaljnjega razvoja.

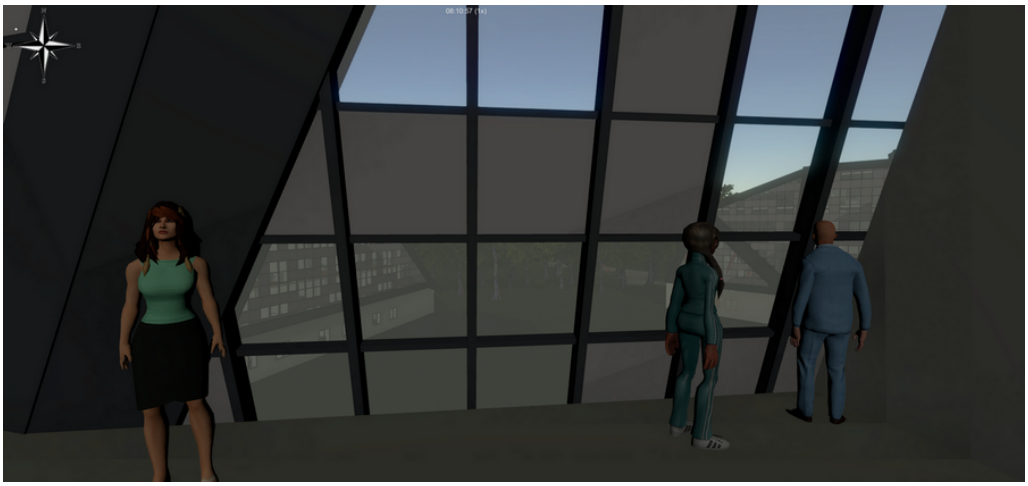
6.1 Interaktivna vizualizacija idejne zasnove novega kulturnega središča v Mariboru

Razvili smo program, ki vizualizira idejno zasnovo kulturnega središča v Mariboru [4] in se bo uporabil kot interaktivni prikaz notranjosti kulturnega kompleksa ter njene okolice. Uporabniku je omogočeno poljubno sprehajanje na sceni okoli kompleksa, približno v radiju 800 m, kar mu omogoča videti arhitekturo iz različnih zornih kotov. Za prikaz smo izbrali pogled iz parka proti severu v jutranjih urah (slika 6.1).



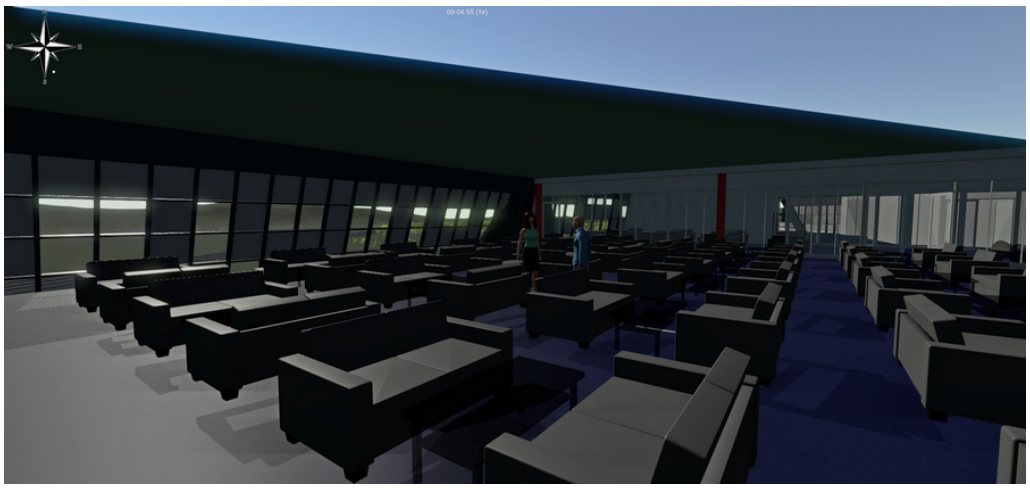
Slika 6.1: Pogled na kulturno središče iz parka

Ljudje so postavljeni na ključnih lokacijah, kjer so statični, služijo pa predvsem za to, da uporabnik aplikacije dobi občutek velikosti prostorov. Nekateri agenti se lahko tudi premikajo po zgradbi, tem lahko sledijo, da odkrivajo nove povezave, služijo pa tudi za večjo realističnost. Slika 6.2 prikazuje pogled iz zgradbe proti parku, ki je na zahodni strani.



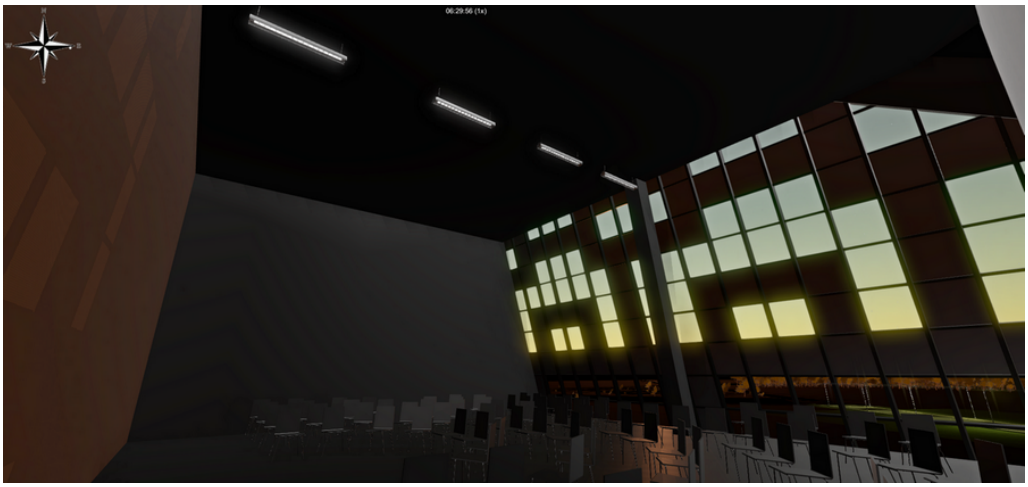
Slika 6.2: Pogled iz pisarne

Ker je prikazana arhitektura zelo obsežna, je po zgradbi postavljenih nekaj interesnih točk na zanimivih lokacijah. Te smo določili vnaprej in jih uporabnik ne more dodajati ali odstranjevati. Točke lahko izbira iz glavnega menija, program pa bo pogled prestavil na izbrano lokacijo. Ena od takih lokacij je restavracija v 5. nadstropju zgradbe, ki jo vidimo ob sončnem vzhodu (slika 6.3).



Slika 6.3: Restavracija ob sončnem vzhodu

Uporabnik lahko uro in hitrost dneva poljubno spreminja, s tem pa tudi opazuje arhitekturo pod različnimi svetlobnimi pogoji. Za primeren prikaz tega smo izbrali vadbeni prostor orkestra. Na stropu je viden niz luči notranje razsvetljave, ki so prižgane, lahko pa jih tudi izklopimo z uporabo stikala na steni. Jutranja svetloba se preliva v prostor, odbija od gladkih tal, na steni pa so prav tako lepo vidne sence konstrukcije obešene fasade (slika 6.4).



Slika 6.4: Vadbišče orkestra s stropno razsvetljavo v jutranji zarji

6.2 Potencialne nadgradnje

6.2.1 Interaktivno vizualno orodje med projektiranjem, za arhitekte, biro

Programi za modeliranje in projektiranje v arhitekturi omogočajo osnovno premikanje oziroma letenje po sceni z namenom, da si objekte lahko ogledujemo iz poljubnih zornih kotov. Arhitekt med delom velikokrat želi preveriti ali predstaviti, kako celoten objekt ali določena podrobnost izgleda iz nekega zornega kota. Velikokrat je ta funkcionalnost minimalna ter je omejena na licenco modelirnega programa. Z dopolnitvijo naše rešitve, bi lahko omogočili kolaborativno sodelovanje preko spletne različice programa.

6.2.2 Predstavitveno orodje za nepremičninske agente

Program lahko uporabimo kot oglasni pripomoček oziroma kot predstavitveno orodje, ki je dostopno preko spleta, mobilne naprave ali namiznega računalnika. Stranka se lahko pred nakupom ali najemom prepriča, ali bo stanovanje ali hiša ustrezala njenim potrebam in zahtevam.

6.2.3 Orodje za opremljanje prostorov

Dopolnjen program bi lahko uporabiili pri prodaji stanovanjske opreme oziroma opremljanju in obnavljanju stanovanja in hiše. Bazo podatkov in 3D modele bi pridobili od proizvajalcev opreme, kot na primer Ikea, Miele, Bosch, Elektroluks. Uporabniki bi lahko uvozili 3D modele svojih stanovanj ali hiš, ki so si jih pripravili v enem od neštetih orodij za izdelavo in planiranje stanovanj.

6.2.4 Navidezni obisk, predstavitev muzejev in podobnih objektov

Muzeji in druge kulturne ustanove bi lahko ponudile navidezne obiske razstav. Ideja se mi je porodila, ko sem gledal v Google street view notranjost muzejev v Berlinski muzejski četrti.

6.2.5 Razne simulacije in vizualizacije

Dopolnjen program bi lahko uporabili za simulacije, kot na primer vizualizacija evakuacije in prepustnosti evakuacijskih poti, v trgovskih centrih, letališčih, šolah, kinematografskih kompleksih, kulturnih središčih itn.

Poglavje 7

Zaključek

Aplikacija ne dosega grafične podrobnosti vizualizacije, kot smo želeli, vendar vseeno zadosti minimalnim potrebam za MVP. Veliko časa se je porabilo na izdelavi dovolj kakovostne mreže terena, kajti, kot smo ugotovili, podrobnosti digitalne elevacije, dobljene na Google Earth, za ta projekt niso bile zadovoljive. Veliko boljši vir je SRTM. Za prepričljivo in podrobno okolico je prav tako potrebno veliko dela v sami predstavljeni zgradbi in okoli nje, hkrati pa smo uporabili zgolj brezplačne gradnike, kar nas je dodatno omejevalo pri izbiri in kakovosti. Agenti (ljudje in vozila) so zelo osnovni, vendar dobro izhodišče za nadaljnji razvoj simulacije gibanja in njihove interakcije z arhitekturo. Veliko časa smo porabili tudi za prilagoditev in uvoz 3D modela zgradbe, zato bi morda bilo smiselno v prihodnje izdelati modul za pretvorbo 3D modela iz ArchiCAD .PLN datoteke v .FBX datoteko, tako da odpravi najbolj pogoste človeške napake ali pomanjkljivosti obstoječih pretvornikov ter da se olajša sam proces uvoza. Testiranje aplikacije in gradnja verzij je potekala s pomočjo servisa Unity Cloud Build, brez katerega ne bi pravočasno identificirali marsikatere težave, kljub temu pa je bil vzorec testerjev in število ponovitev zelo omejeno, zato bi za nadaljnji razvoj morali povečati število testerjev, platform in ponovitev.

Literatura

[1] Mesopotamian City Plan for Nippur.
Dosegljivo: http://cartographic-images.net/Cartographic_Images/101_Mesopotamian_City_Plan,_Nippur.html.
(Dostopano 30. 8. 2016).

[2] Forma Urbis Romae.
Dosegljivo: http://formaurbis.stanford.edu/docs/FURmap.html.
(Dostopano 30. 8. 2016).

[3] Filipo Brunelleschi.
Dosegljivo: https://en.wikipedia.org/wiki/Filippo_Brunelleschi.
(Dostopano 30. 8. 2016).

[4] M. Hausmeister, *Idejna zasnova kulturnega parka z kulturnim centrom v namene revitalizacije območja na vzhodnem robu mesta Maribor*, diplomska naloga, Fakulteta za arhitekturo, Univerza v Ljubljani, 2016.

[5] SRTM v4.1.
Dosegljivo: http://www.cgiar-csi.org/data/srtm-90m-digital-elevation-database-v4-1
(Dostopano 30. 8. 2016).

[6] S. Boeykens, *Unity for Architectural Visualization*, Packt Publishing, 2013.

[7] R. Barrera, *Unity AI Game Programming – Second Edition*, Packt Publishing, 2015.

[8] A. Thorn, *Unity Animation Essentials*, Packt Publishing, 2015.

[9] M. Szczesnik, *Unity 5.x Animation Cookbook*, Packt Publishing, 2016.

[10] S. Jackson, *Unity 3D UI Essentials*, Packt Publishing, 2015.